IN THE CLAIMS

The pending unamended claims are reproduced below:

1. (PREVIOUSLY PRESENTED) A method for checking a version of an abstract data type stored in a database, comprising:

(a) constructing an identifier for the abstract data type, wherein the identifier comprises a concatenation of various attributes for the abstract data type that is substantially unique to the abstract data type;

(b) hashing the constructed identifier to generate a signature hash value for the abstract data type;

(c) storing the signature hash value in the database and a class definition for the abstract data type; and

(d) comparing the signature hash value from the database with the signature hash value from the class definition after the class definition is instantiated in order to verify that the class definition is not outdated.

2. (ORIGINAL) The method of claim 1, wherein the comparing step comprise:

(1) instantiating the class definition as a library function;

(2) accessing the abstract data type via the library function; and

(3) comparing the signature hash value from the database and the signature hash value from the class definition in order to verify that the class definition is not outdated.

3. (ORIGINAL) The method of claim 1, wherein the class definition is outdated when the abstract data type has been altered without the signature hash value being re-generated and re-stored in the database and the class definition.

4. (ORIGINAL) The method of claim 1, wherein the abstract data type is stored in a relational database.

5. (CANCELED)

6. (PREVIOUSLY PRESENTED) A method for generating a signature hash value for checking a version of an abstract data type stored in databases, comprising:

(a) constructing an identifier for the abstract data type, wherein the identifier comprises a concatenation of various attributes for the abstract data type that is substantially unique to the abstract data type;

(b) hashing the constructed identifier to generate a signature hash value for the abstract data type; and

(c) storing the signature hash value in the database and a class for the abstract data type.

7. (CANCELED)

8. (PREVIOUSLY PRESENTED) A method for verifying a signature hash value for checking a version of an abstract data type stored in a database, comprising:

(a) accessing a first signature hash value from the database and a second signature hash value from a class definition for the abstract data type, wherein the first and second signature hash values have been constructed from an identifier for the abstract data type, the identifier comprises a concatenation of various attributes for the abstract data type that is substantially unique to the abstract data type, and the identifier has been hashed to generate the first and second signature hash values; and

(b) comparing the first signature hash value with the second signature hash value after the class definition is instantiated in order to verify that the class definition is not outdated.

9. (ORIGINAL) The method of claim 8, wherein the accessing step comprise:

(1) instantiating the class definition as a library function; and

(2) accessing the abstract data type via the library function.

10. (ORIGINAL) The method of claim 8, wherein the class definition is outdated when the abstract data type has been altered without the signature hash value being re-generated and re-stored in the database and the class definition.

11. (CANCELED)

12. (PREVIOUSLY PRESENTED) An apparatus for checking a version of an abstract data type stored in a database, comprising:

(a) means for constructing an identifier for the abstract data type, wherein the identifier comprises a concatenation of various attributes for the abstract data type that is substantially unique to the abstract data type;

(b) means for hashing the constructed identifier to generate a signature hash value for the abstract data type;

(c) means for storing the signature hash value in the database and a class definition for the abstract data type; and

(d) means for comparing the signature hash value from the database and the signature hash value from the class definition after the class definition is instantiated in order to verify that the class definition is not outdated.

13. (ORIGINAL) The apparatus of claim 12, wherein the means for comparing comprise:

(1) means for instantiating the class definition as a library function;

(2) means for accessing the abstract data type via the library function; and

(3) means for comparing the signature hash value from the database and the signature hash value from the class definition in order to verify that the class definition is not outdated.

14. (ORIGINAL) The apparatus of claim 12, wherein the class definition is outdated when the abstract data type has been altered without the signature hash value being re-generated and re-stored in the database and the class definition.

15. (PREVIOUSLY PRESENTED) The apparatus of claim 12, wherein the abstract data type is stored in a relational database.

16. (CANCELED)

17. (PREVIOUSLY PRESENTED) An apparatus for generating a signature hash value for checking a version of an abstract data type stored in databases, comprising:

(a) means for constructing an identifier for the abstract data type, wherein the identifier comprises a concatenation of various attributes for the abstract data type that is substantially unique to the abstract data type;

(b) means for hashing the constructed identifier to generate a signature hash value for the abstract data type; and

(c) means for storing the signature hash value in the database and a class definition for the abstract data type.

18. (CANCELED)

19. (PREVIOUSLY PRESENTED) An apparatus for verifying a signature hash value for checking a version of an abstract data type stored in a database, comprising:

(a) means for accessing a first signature hash value from the database and a second signature hash value from a class definition for the abstract data type, wherein an identifier is constructed for the abstract data type, the identifier comprises a concatenation of various attributes for the abstract data type that is substantially unique to the abstract data type, and the identifier is hashed to generate the first and second signature hash values; and

(b) means for comparing the first signature hash value with the second signature hash value after the class definition is instantiated in order to verify that the class definition is not outdated.

20. (ORIGINAL) The apparatus of claim 19, wherein the means for accessing comprise:

(1) means for instantiating the class definition as a library function; and

(2) means for accessing the abstract data type via the library function.

21. (ORIGINAL) The apparatus of claim 19, wherein the class definition is outdated when the abstract data type has been altered without the signature hash value being re-generated and re-stored in the database and the class definition.

22. (CANCELED)

23. (PREVIOUSLY PRESENTED) An article of manufacture embodying logic for performing a method for checking a version of an abstract data type stored in a database, the method comprising:

(a) constructing an identifier for the abstract data type, wherein the identifier comprises a concatenation of various attributes for the abstract data type that is substantially unique to the abstract data type;

(b) hashing the constructed identifier to generate a signature hash value for the abstract data type;

(c) storing the signature hash value in the database and a class definition for the abstract data type; and

(d) comparing the signature hash value from the abstract data type and the signature hash value stored from the class definition after the class definition is instantiated in order to verify that the class definition is not outdated.

24. (ORIGINAL) The method of claim 23, wherein the comparing step comprise:

(1) instantiating the class definition as a library function;

(2) accessing the abstract data type via the library function; and

(3) comparing the signature hash value from the database and the signature hash value from the class definition in order to verify that the class definition is not outdated.

25. (ORIGINAL) The method of claim 23, wherein the class definition is outdated when the abstract data type has been altered without the signature hash value being re-generated and re-stored in the database and the class definition.

26. (ORIGINAL) The method of claim 23, wherein the abstract data type is stored in a relational database.

27. (CANCELED)

28. (PREVIOUSLY PRESENTED) An article of manufacture embodying logic for performing a method for generating a signature hash value for checking a version of an abstract data type stored in databases, comprising:

(a) constructing an identifier for the abstract data type, wherein the identifier comprises a concatenation of various attributes for the abstract data type that is substantially unique to the abstract data type;

(b) hashing the constructed identifier to generate a signature hash value for the abstract data type; and

(c) storing the signature hash value in the database and a class definition for the abstract data type.

29. (CANCELED)

30. (PREVIOUSLY PRESENTED) An article of manufacture embodying logic for performing a method for verifying a signature hash value for checking a version of an abstract data type stored in a database, comprising:

(a) accessing a first signature hash value from the database and a second signature hash value from a class definition for the abstract data type, wherein an identifier is constructed for the abstract data type, the identifier comprises a concatenation of various attributes for the abstract data type that is substantially unique to the abstract data type, and the identifier is hashed to generate the first and second signature hash values; and

(b) comparing the first signature hash value with the second signature hash value after the class definition is instantiated in order to verify that the class definition is not outdated.

31. (ORIGINAL) The method of claim 30, wherein the accessing step comprise:

(1) instantiating the class definition as a library function; and

(2) accessing the abstract data type via the library function.

32. (ORIGINAL) The method of claim 30, wherein the class definition is outdated when the abstract data type has been altered without the signature hash value being re-generated and re-stored in the database and the class definition.

33. (CANCELED)

34. (PREVIOUSLY PRESENTED) At least one data structure stored in a memory for use

in checking a version of an abstract data type stored in a database, the data structure comprising a signature hash value for the abstract data type generated from an identifier constructed for the abstract data type, wherein the identifier comprises a concatenation of various attributes for the abstract data type that is substantially unique to the abstract data type and the identifier is hashed to generate the signature hash value for the abstract data type.

35. (ORIGINAL) The data structure of claim 34, wherein the signature hash value is stored in the database.

36. (ORIGINAL) The data structure of claim 35, wherein the signature hash value is stored in a class definition for the abstract data type.

37. (ORIGINAL) The data structure of claim 35, wherein the signature hash value from the database is compared to the signature hash value from the class definition after the class definition is instantiated in order to verify that the class definition is not outdated.

38. (ORIGINAL) The data structure of claim 37, wherein the class definition is outdated when the abstract data type has been altered without the signature hash value being re-generated and re-stored in the database and the class definition.

39. (ORIGINAL) The data structure of claim 34, wherein the abstract data type is stored in a relational database.

40. (CANCELED)